NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CONSTRUCTING HIGHER-ORDER DE BRUIJN GRAPHS

by

D'Hania J. Hunt

June 2002

Thesis Advisor:	Harold Fredricksen
Second Reader:	Craig W. Rasmussen

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Constructing Higher-Order de Bruijn Graphs			**5. FUNDING NUMBERS**
6. AUTHOR(S) CPT D'Hania J. Hunt			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release, Distribution is Unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

We construct binary de Bruijn graphs of odd order using recursive generation. We also explore the properties and nuances of these particular graphs. The recursive method developed for this thesis could in principle be used for other de Bruijn graphs of a different order. Suggestions on how this is accomplished are included in the paper and areas of further research topics.

14. SUBJECT TERMS Graph Drawing, Recursion, de Bruijn Graphs			**15. NUMBER OF PAGES** 61
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CONSTRUCTING HIGHER-ORDER DE BRUIJN GRAPHS

D'Hania J. Hunt
Captain, United States Army
B.S., United States Military Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author:	D'Hania J. Hunt
Approved by:	Harold Fredricksen Thesis Advisor
	Craig W. Rasmussen Second Reader
	Clyde W. Scandrett Chairman, Department of Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We construct binary de Bruijn graphs of odd order using recursive generation. We also explore the properties and nuances of these particular graphs. The recursive method developed for this thesis could in principle be used for other de Bruijn graphs of a different order. Suggestions on how this is accomplished are included in the paper and areas of further research topics.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author wishes to thank Professor Harold Fredricksen for his guidance, patience and support throughout this process. Thanks for introducing to me a whole new area of mathematics. Thanks to Professor Craig Rasmussen for being the second reader.

The author would also like to acknowledge the support of LTC(R) Willie Hunt, Dianne Hunt, Wesley Hunt, Wrendon Hunt and Harry Hung, thanks for all of your prayers and love.

Lastly and most importantly the author acknowledges the Lord, Jesus Christ, for the blessing of patience and perseverance. Thank the Lord it is done!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A graph is a discrete structure consisting of a set of nodes and a relation on that set that is conveniently visualized as a set of edges joining certain pairs of nodes. We make relatively little use of the terminology and theory of graphs here; what we use will be defined as the need arises. In some applications of graph theory to practical problems (e.g., VLSI circuit design, software engineering diagrams, and the depiction of graphs for purposes of describing graph algorithms) it is useful to have a method for representing a graph that in one or another way facilitates understanding. The goal might be to understand the structure of the object modeled by the graph itself, or to understand the behavior of an algorithm whose input includes a graph or network. The study of models, algorithms, and systems for visualization of graphs and networks is called *graph drawing*. For a survey of the field and its applications, see Di Battista (1999).

Any sequence with the property that all but some small number of its terms are defined by some rule applied to their predecessors is said to be *recursively defined*. The use of "term" here is broadly defined; the sequence might be of numbers, of functions, or (as in this paper) of graphs. Similarly, an algorithm is said to be recursive if, in the course of solving a problem of size n, the algorithm calls itself to solve a smaller problem. For a gentle introduction to recursion, see a standard introductory text on discrete mathematics, such as Rosen (1999).

The focus of this paper is the design of a recursive graph-drawing algorithm for generating drawings of a particular sequence of graphs known as the Good-de Bruijn graphs.

A. BACKGROUND

Good (1946) and de Bruijn (1946) independently created the Good-de Bruijn graphs to solve the problems of the existence and enumeration of certain cycles of 0s and 1s, namely cycles of length 2^n containing each binary n-tuple. Small versions of the graph were easily drawn to illustrate their ideas, but larger versions of the graph proved unwieldy to draw. Massey and Liu (1965) to emphasize certain properties drew alternate versions of the graph. In particular, the Massey-Liu graph is easily extended recursively to larger sizes, something that is not apparent as originally depicted by Good and de Bruijn.

There are additional interesting problems concerning the subject of shift register sequences and properties of the associated de Bruijn graphs that arise only because of the existence of the graph model. One example of such a problem is Golomb's Conjecture (1967) on the number of disjoint cycles that can simultaneously occur in the Good-de Bruijn graph. Several papers have appeared concerning this conjecture, including one by Lempel (1971) describing both another conjecture that implies Golomb's and also a structure that would have to exist in the graph if Lempel's Conjecture were valid. With the proof of the Golomb-Lempel Conjecture by Mykkeltveit (1971), the Cycle Adjacency Array (CAA) described by Lempel gives rise to another way to describe the de Bruijn graph. This then leads to additional interesting questions. The graph is not easily extendible in this configuration. Nevertheless, larger versions of the graph than had previously appeared aren't too difficult to construct. Each different version of the graph exhibits its own specific properties better than another and each new presentation provides additional suggestions for new research topics. With all of this information at hand, we set out to find a streamlined method to construct higher-order binary de Bruijn graphs via a recursive generation of the graphs.

B. GOALS

This thesis defines a recursive process to construct higher-order de Bruijn graphs. The process suggests a (seeming) fractal property that may appear in the graph. By this process it is easier to build the graphs and eventually provide insight from a visual inspection of the graphs. In the paper we present de Bruijn graphs of sizes up to 8192 nodes. The purpose of this set of graphs is to show properties of the graph on a small scale and to demonstrate the ability to recursively build higher-order graphs.

C. ORGANIZATION OF STUDY

The paper is organized into five chapters. The first chapter is the introduction. Chapter II focuses on previous research in the development of de Bruijn graphs. More detail is given there on the conjectures and graphs mentioned in Chapter I. Chapter III describes the process used to recursively build the odd order binary de Bruijn graphs and defines constraints and parameters of their construction; construction of even order graphs is similarly definable. Chapter IV illustrates the results from the outlined constructive procedures. Chapter V gives conclusions, recommendations and areas for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

The de Bruijn graph was developed to provide a model for the solution to the problem of finding a cycle of length 2^n containing every binary n-tuple (de Bruijn, 1946), (Good, 1946). The graph B_n, of span n, contains 2^n nodes (labeled with the binary n-tuples) and 2^{n+1} arcs. The term span is defined as the length of the shift register, the size of the graph. There is an arc from the node $x = x_1x_2\ldots x_n$ to the node $y = y_1y_2\ldots y_n$ if and only if $x_2x_3\ldots x_n = y_1y_2\ldots y_{n-1}$. The solution to the problem is then given as a Hamiltonian cycle in the span-n graph visiting each node exactly once. The number of such cycles is also known (de Bruijn, 1946) as well as the existence of cycles of each length 1, 2,…, 2^n (Golomb, 1967). The number of de Bruijn cycles is found employing a "doubling" of the graph, i.e., an Euler circuit visiting each edge in the span-(n-1) graph is equivalent to a Hamiltonian cycle in the span n-graph. This implies a recursive generation of the graphs, although actually drawing the graphs is difficult. Krahn (1994) describes a generalization of these sequences for paths that cover the edges of B_n more than once. Other properties of the graph include its 2-regularity, 3-color ability (Berge, 1962) and non-planarity (in general) (Johnson, 1970). For readers interested in further study of de Bruijn sequences see Fredricksen (1982).

Since drawing the graph for higher orders is arduous, Massey (1965) developed a modularly recursive, alternative version of the graph. This version is easily extendable to larger sizes. A depiction of the span-4 graph is shown in Figure 2.1. Note that the labeled regions depict the decimal representations of the nodes in the graph. The (implicit) arcs are described as coming from internal nodes to the nodes immediately exterior to them. There is also an arc from node 0 to itself. The outer "ring" only appears to describe the arcs that also appear from $8 \rightarrow 0$ and $8 \rightarrow 1$; $9 \rightarrow 2$ and $9 \rightarrow 3$, etc. If the entries in the (phantom) outer ring are each increased by 16, the span-4 graph with its outer ring depicts the span 5-graph. The addition of the appropriate phantom ring $0' \rightarrow 31'$ then describes the additional induced arcs of the span 5-graph to the interior nodes.

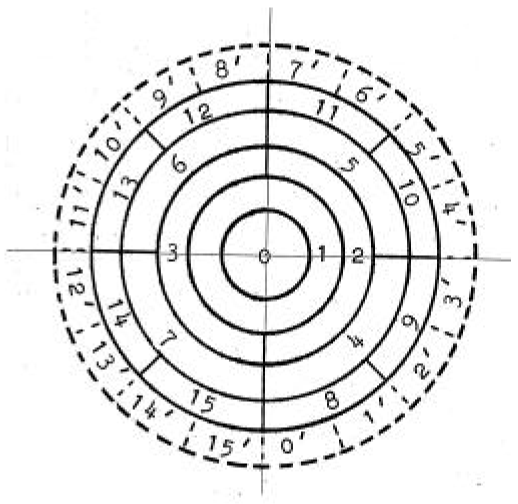


Figure 2.1. Massey-Liu Graph for span n = 4

Previously, graphs B_n were typically drawn exhibiting a left-right symmetry ($x_1 \ldots x_n$ versus $x_n \ldots x_1$) with respect to a vertical centerline. See Figures 2.2 and 2.3 for versions of the graph B_n, for $1 \leq n \leq 4$. Nodes that are self-symmetric appear on the centerline. There is also a top-bottom symmetry ($x_1 \ldots x_n$ versus $\overline{x_1} \ldots \overline{x_n}$) with respect to the center point of the graph; here $\overline{x}$ denotes the binary complement of x. Graphs larger than B_5 are complex to draw.

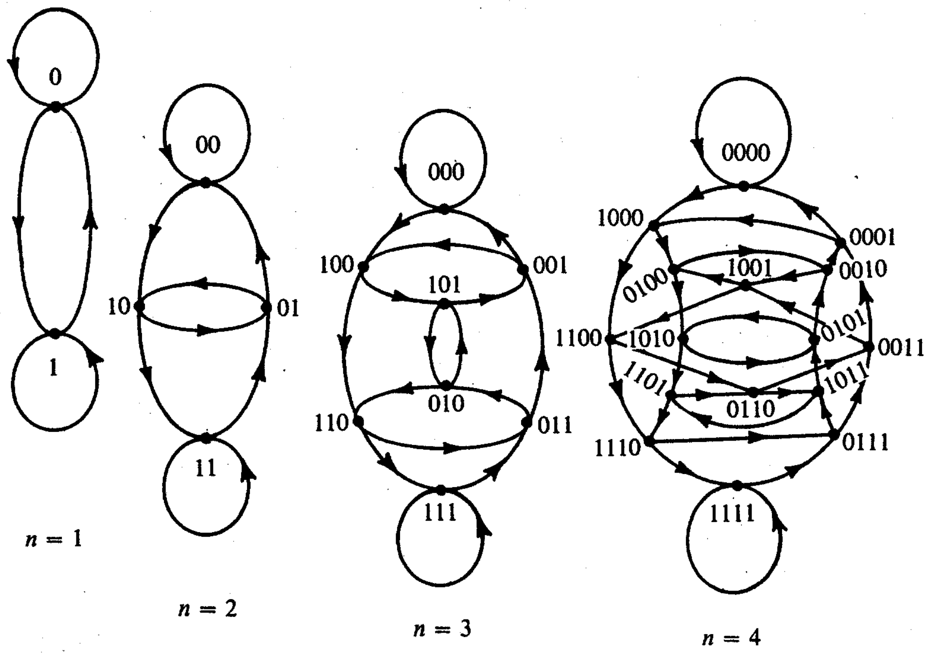


Figure 2.2. B_1-B_4 as depicted by Golomb (1967)

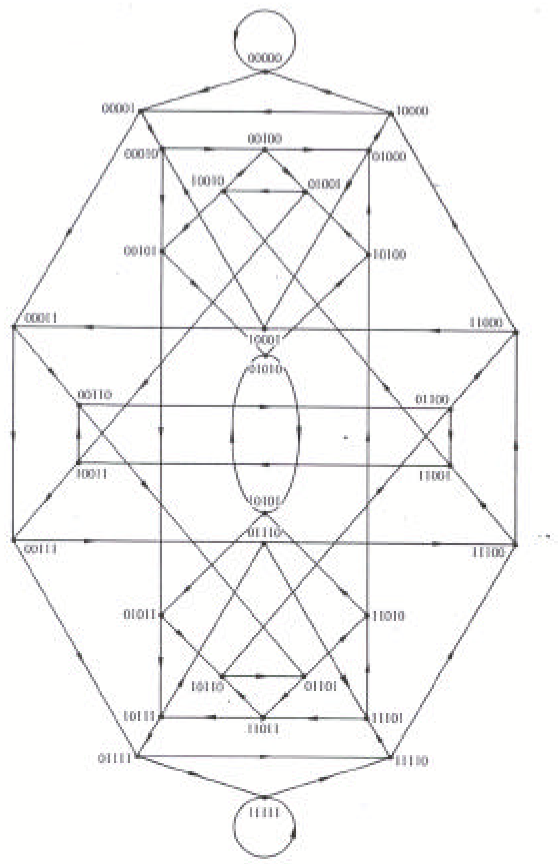


Figure 2.3. B_5 as depicted by Golomb (1967)

Employing the notion of doubling, Taylor (2001) creates a copy of B_6. In Figure 2.4, we show a depiction of this B_6. Note that the nodes are named by their decimal equivalent values.

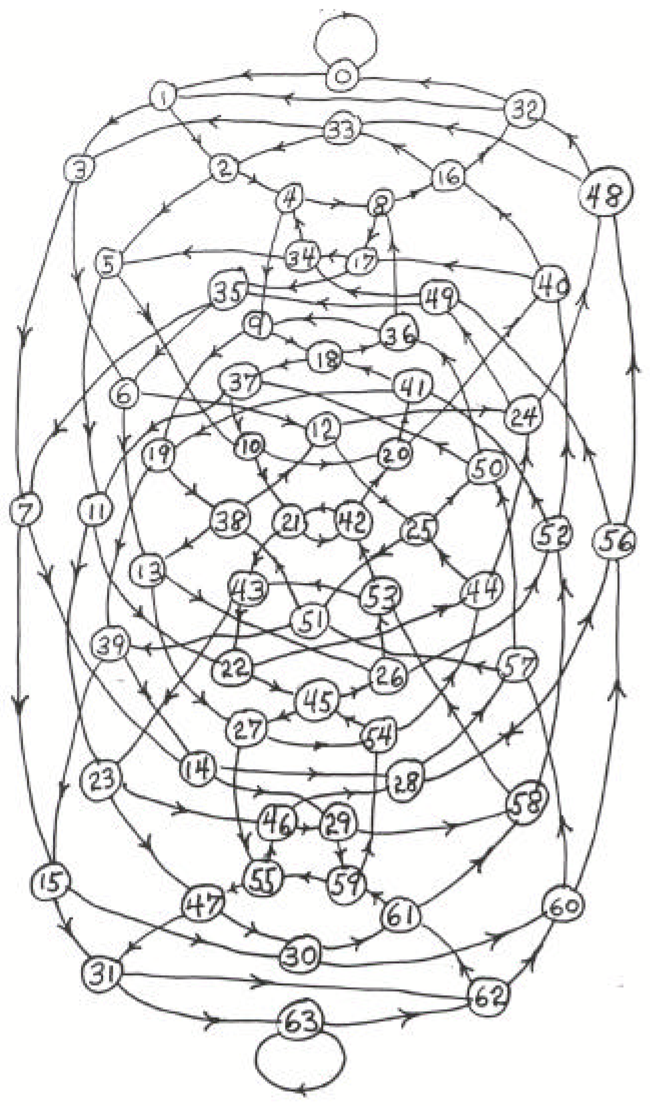


Figure 2.4. Taylor’s depiction of de Bruijn graph for n = 6

Fredricksen (1992) displayed the graph as embedded in a sphere, shown in Figure 2.5. Note that as in the Massey-Liu graph, Figure 2.1, all the arcs do not appear. One successor arc is drawn from x→ y as it would appear in the pure-cycling shift register (Golomb, 1967). The alternative arc (not appearing) would be drawn from the predecessor node x to the alternate successor node y ⊕ 00…01, where ⊕ denotes a modulo 2 addition of the n-long string 00…01.

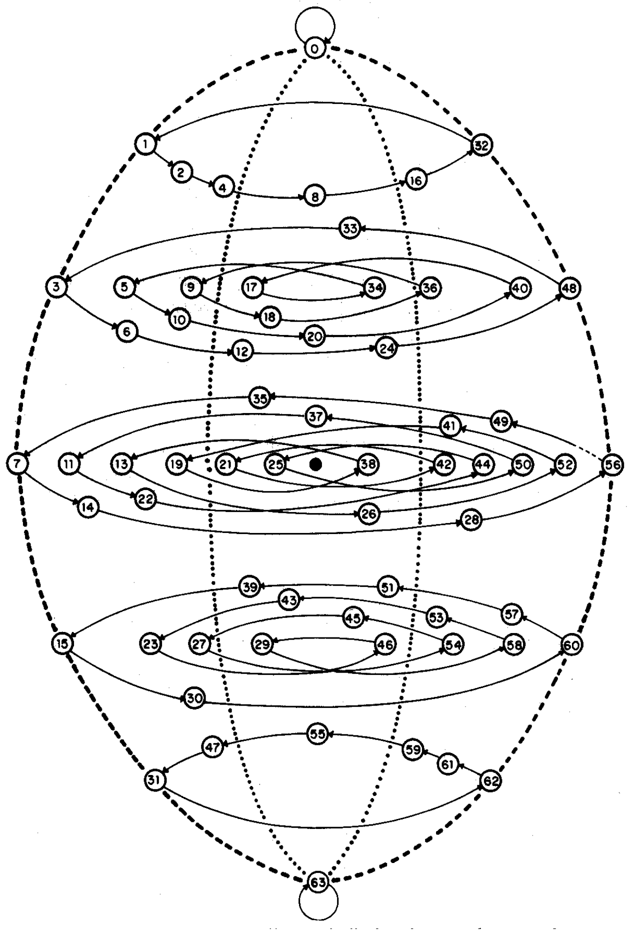


Figure 2.5. Good-de Bruijn graph of span n = 6 embedded in a sphere

We note that the de Bruijn graph is an important model in the areas of Markov Modeling, Convolutional Codes, Trellis Coding and Sequential Decoding. The Viterbi Algorithm, which finds the most likely path through a graph given a maximum-likelihood measure on code words defined by the Convolutional Code, also uses the de Bruijn graph. An implementation of the Viterbi Algorithm in silicon led Collins, et al. (1992) to discover the largest planar subgraphs of the graphs B_6 and B_7 (of 64 and 128 nodes, respectively) and they then extended these into a span 14 (B_{14}) Viterbi decoder of 16,384 nodes, which flew on the Galileo Spacecraft.

The model of the graph suggests many research problems. One of these, suggested by Golomb (1967), is the conjectured largest number of simultaneous disjoint cycles in the graph. The conjectured maximum is given by $Z(n) = ?n ?_{d/n} \Phi(d)\, 2^{n/d}$, where the summation is taken over all divisors d of n and Φ is Euler’s totient function. The sum counts all necklaces of length n in 2 colors of beads or, equivalently, enumerates the equivalence classes of binary n-tuples under cyclic rotation. Lempel (1971) conjectured that this same number of vertices, if removed from the graph, would be sufficient to leave the (directed) subgraph acyclic. Mykkeltveit (1971) proved this. Lempel’s conjecture implied Golomb’s and inferred the existence of a Cycle Adjacency Array (CAA).

In the CAA, the set S of removed nodes is subjected to the transformations L, L^2, etc. to form a sequence, S, L(S),…,$L^k(S) = S$, which is ultimately periodic. The transformations is defined by $L(S) = \{x|\ x \varepsilon S$ or 2x and 2x+1 whenever both x and $x+2^{n-1}\ \varepsilon S\}$. Thus, x and its companion $x+2^{n-1}$ appearing in S are replaced by the two possible (shared) successors 2x and 2x+1, respectively, in L(S). Note, these changes $x \rightarrow 2x$ and $x+2^{n-1} \rightarrow 2x+1$ describe one step along the cycles defined by the cyclic equivalence classes described earlier. An example of a CAA for the case n = 5 is shown in Figure 2.6.

0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	2	4	4	4	4	4	4	4	4	8	16
17	17	3	6	6	6	12	12	12	24	24	24	17
18	18	18	5	10	10	10	10	10	10	20	9	18
19	19	19	7	7	7	14	14	28	25	25	25	19
21	21	21	21	11	22	13	13	13	26	21	21	21
27	27	27	27	27	23	15	30	29	27	27	27	27
31	31	31	31	31	31	31	31	31	31	31	31	31

Figure 2.6. Cycle Adjacency Array for n = 5

The first column to the left of the vertical bar is the removed set S. The successive columns are $\Gamma(S)$, $\Gamma^2(S)$, until $\Gamma^{12}(S) = S$ again. When the set S has been chosen properly and $\Gamma^k(S) = S$ with $k > 1$ then the set S satisfies Lempel's conjecture and thereby Golomb's conjecture. As drawn in Figure 2.6, removing the black values and leaving only the red numbers gives a different version of the graph B_5. The numbers are the decimal equivalents of the nodes. The rows are the cycles of the cyclic equivalence classes and appearance on a row is evidence of an adjacency in the graph B_5. The other implicit arc of the node x goes to the node $2x + 1 (\bmod 2^n)$ in the same column of the successor $2x (\bmod 2^n)$ shown in the respective row. We suppress these arcs only to keep the graph easy to view. Larger graphs are possible, such as for $n = 7$ and $n = 9$ as they appear in Fredricksen (1992). One can even suppress the numbers, as they can be understood also!

In another application, Bryant et al. (1991) show that when additional nodes are removed from B_n, maximum independent sets can be formed. Their procedure leads to a fractal-like property in the graph B_n. These various methods to explain the graph and its properties have led to this paper and a method of construction for the graph that is recursive and therefore extendable to larger sizes. A detailed description of our method is given in Chapters III and IV. Thus the research in the underlying combinatorics leads to improvements in the model on occasion, and also the different versions of the model can lead to new mathematical research.

THIS PAGE INTENTIONALLY LEFT BLANK

III. CONSTRUCTING THE GRAPH

The de Bruijn graph and some of its properties and applications are detailed in Chapter II. The difficulty in drawing the graph for larger spans is the inspiration for our efforts to find a better method. Some of the symmetric properties used to draw the graph originally yield the inferences that we need, namely:

1) The nodes of the directed graph B_n, represented by the 2^n binary n-tuples, obey a left-right symmetry with respect to the centerline of the graph (CL_n) so that the node $x_1x_2\ldots x_n$ is symmetrically placed opposite the node $x_nx_{n-1}\ldots x_2x_1$.

2) The nodes of B_n are placed so that the node $x_1x_2\ldots x_n$ is placed symmetrically opposite the node $\overline{x_1}\,\overline{x_2}\ldots\overline{x_n}$ with respect to the center point of the drawing.

When constructing higher-order binary de Bruijn graphs by recursion, several techniques can be considered. It is natural to consider recursive processes, $B_n \rightarrow B_{n+1}$, $B_n \rightarrow B_{n+2}$, $B_n \rightarrow B_{n+3}$, etc., with B_1 forming a basis for the process. Each step increment, n ? n+k, dictates a different recursive method. The step increments also prescribe the number of copies needed for the recursion. That is to say, to implement $B_n \rightarrow B_{n+1}$ requires two copies of B_n, $B_n \rightarrow B_{n+2}$ requires four copies of B_n, $B_n \rightarrow B_{n+3}$ requires eight copies of B_n, etc. Each incremental method also has its own separate and unique challenges and varied nuances of construction. For the purpose of this paper, we focus on one method of recursion, namely $B_n \rightarrow B_{n+2}$. We note that there is a difference between graphs of even and odd order further detailed in Chapter V. This leads to additional issues when applying $B_n \rightarrow B_{n+1}$, $B_n \rightarrow B_{n+3}$, etc. Furthermore, our presentation only addresses graphs of odd order. Even-to-even constructions, even-to-odd, odd-to-even and other recursive methods are topics for future research and development and we describe some of the issues involved in Chapter V. These processes present no *essentially* difficult problems beyond those which we deal with here. Thus, the rest of the paper illustrates odd order binary de Bruijn graphs built by quadrupling, $B_n \rightarrow B_{n+2}$, employing heavily the symmetry properties (1) and (2) above.

A. LABELING

We first describe our labeling conventions. The nodes of the graph are classically labeled by binary n-tuples or their decimal equivalents. We use a binary representation for the nodes, as that fits our construction better. The basis of the recursive process is the graph B_1. Labeling by binary 1-tuples, the top node is labeled 0 and the bottom node is labeled 1. B_1 is shown in Figure 3.1. Note that we have suppressed all of the arcs as they are not relevant for our process. We also include the centerline of symmetry from property 1) and the center point x of symmetry from property 2). These provide an aid to a recursive construction process. Often the center point x will not appear explicitly in our construction. We call the centerline of B_n, CL_n and the center point CP_n.

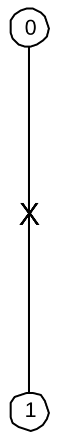


Figure 3.1. de Bruijn Graph, B_1

For each iteration, as the span n increases by 2, the number of nodes appearing increase by a factor of 2^2 or 4. Our labeling system plays a large role in determining the overall graph. The binary strings representing the nodes of B_n are used to implement left-right symmetry and top-bottom symmetry properties described above. They are applied on the nodes of B_n and those of B_{n+2} as we proceed. This binary representation is also used to identify the self-symmetric nodes that belong on the centerline of the respective graphs. Such nodes are self-symmetric if the binary strings are left-right palindromes. This placement of nodes is addressed and clarified in Chapter IV after the recursive process is explained further below.

B. PARAMETERS AND CONSTRAINTS

Before describing the recursive process, we need to establish additional rules for the recursive construction. We refer to these rules as the constraints and parameters of the process. Without fixed constraints and parameters, the de Bruijn graph will assume any arbitrary shape and consequently defeat our purpose of establishing an easy-to-extend pattern.

We define β to be the clockwise measure of the angle between the centerline CL_n and the node 00…01. Beginning with B_1 we establish a target size B_{2m+1} as the end of our process. The constraints demand that the value β of the final graph, B_{2m+1}, does not exceed 60^o. If the initial angle on B_3 is α, see Figure 3.2, then the size of α cannot exceed β/m, according to the recursive process used to build higher order graphs. This point is clarified later in this section. Parameters of our construction also include the length A, the length of the first centerline CL_1 on B_1 and the length B, the distance from the top node 000 on B_3 to the grid center point CP_3, the length C, the distance between the top center node, 010 on B_3, and CP_3. The angle α remains constant throughout the entire recursive process and is constrained as described above. The parameters are indicated in Figure 3.2. Three of the four copies required to complete B_3 are depicted. Note that the distance A is retained on each of the copies used in the figure and the angle α is the same for each of the three copies shown. Details on the colors used and the names of the nodes, etc., are reserved to Chapter IV.

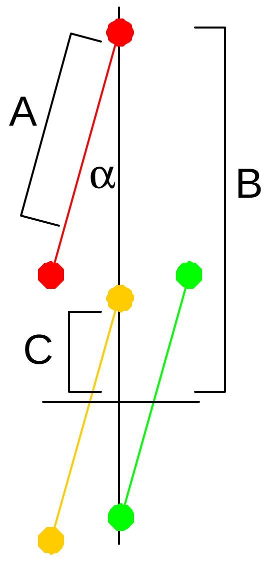


Figure 3.2. Parameters for our recursive process

By adjusting the parameters α, A, B and C, as illustrated on the graph B_3, we change the shape of the "building block". B_3 is the "building block" for larger graphs in our illustration of the process. The shape of B_3 dictates the shape of all subsequent graphs. As the lengths A and B increase, the entire graph B_{2m+1} gets longer, and as these lengths decrease the graph gets shorter. How C affects the graph depends upon the relative adjustments to A and B. Increasing α widens the graph and decreasing it makes the graph thinner.

The chosen sizes of the parameters are determined in the following way. Length A is established on B_1. This is an overall scaling parameter and has little effect in the sequel. Lengths B and C will be discussed later. It is best to describe their choice in the recursive process section. The measure of α is dictated by a predetermined span of the final graph as described above. That is, the choice of α is made with an eye to the ultimate target value of B_n that we seek to achieve. Namely, to produce B_9 by B_1 ? B_3 ? B_5 ? B_7 ? B_9, we perform three iterations of $B_n \rightarrow B_{n+2}$, since the basis for the process is really B_3. But if the final angle measure satisfies $\beta \leq 60^o$, then the initial angle α must have measure satisfying $\alpha \leq \beta/4 \leq 60^o/4$, therefore $\alpha \leq 15^o$. Further discussion of the other parameters is incorporated in the recursive process described next.

C. RECURSIVE GENERATION

The recursive process is a set of instructions to be repeated at each iteration. In designing the process, we have in mind that it should be easy to follow and easy to duplicate. Moreover, the target graph B_n should be easy to understand and view. The process to build B_3 utilizes B_1 of Figure 3.1, rotates it clockwise by α from a centerline, and four copies are produced. Parameter A and angle α have been described above. We now describe how the parameters B and C are chosen. Details of the names of the nodes as described are given later in this section. B is the length from the top node, 000, of the first copy of B_1 to the center point CP_3 and C is the length between the top center node, 010, and the center point. Clearly the lengths A, B and C are inter-dependent. Namely, adjustments on length B automatically adjust length C and vice-versa according to the symmetry properties 1) and 2). In a similar way adjustments on length A result in adjustments affecting lengths B and C, though not necessarily in the same manner.

The following choices affect equivalent ways of adjusting B and C and ultimately completing the graph:

a) From the labeling, we know that nodes 010 and 101 on B_3 are self-symmetric and therefore appear on CL_3. Note that node 010 should be placed above node 101, as our experience seems to say.

b) CP_3 can be determined on CL_3, midway between nodes 010 and 101. Thus lengths B and C are implicitly determined.

c) The location of node 111 on CL_3 is determined by symmetry (2) as symmetrically opposite node 000, with respect to the center point CP_3.

d) The location of node 011 below node 001 on a line parallel to CL_3 is determined by symmetries (1) and (2) on node 100. Equivalently, we place node 110 below node 100 on another line parallel to the centerline.

Following the symmetric properties, the construction results in the skeleton of B_3 (without arcs, of course). From here we begin the recursive generation, proper. To build B_5 we require four copies of B_3. The nodes of B_3 are labeled as 000, 001, 010, 011, 100, 101, 110, 111. The first copy of B_3 has label 00 appended on each node as 00(_ _ _) for placement on B_5. Subsequent copies are labeled as 01(_ _ _), 10(_ _ _), and 11(_ _ _) for the four copies. These are rotated and translated according to a procedure delineated further in Chapter IV. The symmetry properties (1) and (2) are employed heavily in the translations applied to the four copies. The CL_3 of each copy of B_3 plays an important role in the construction of B_5. Surprisingly, the construction of B_5 depends in a crucial way on the eventual CL_7 that will be established on B_7. The nodes that appear on CL_7 are assumed to be properly placed in B_5 only if they lie on a line in B_5 according to our procedure. However, their location on that line depends in a surprising way on the location of the nodes on CL_3! This is explained in more detail in Chapter IV. Here we merely note our process and self-imposed rules. Namely, given B_n, we rotate it clockwise α, make four copies and place these on the grid according to symmetric properties (1) and (2).

In Chapter IV the names of the nodes used are typically suppressed and a color-coding system is used to describe the nodes and the binary representation for the nodes for the four copies of B_n used. Once these copies are placed on the graph the construction is complete. This describes the recursive method for a $B_n \rightarrow B_{n+2}$ construction for odd values of n. For even values of n a different basis is required as described in Chapter V.

The same procedure is used to build graphs of arbitrary order. We see in Chapter IV that, during the construction, the parameters A, B and C determining the location of the nodes of B_{n+2} might need to be adjusted. These adjustments all take place on the “building block”, B_3. Thus, the building block B_3 is crucial to our process and we need to build a “reasonable” B_3. By reasonable we mean that no overlapping nodes should appear in the first few iterations of the process. So if B_7 has overlapping nodes we return to B_3 and adjust the parameters B and C. The impact of the adjustments on B or C might be earliest seen after several iterations, so it is typically necessary to draw B_7 to ensure that B_3 is going to produce a “reasonable” graph. It is unrealistic to think that the nodes will never overlap, except in principle. In Chapter IV, we see that it becomes inevitable that nodes overlap beyond a certain value of the span.

Now that we have procedures for building odd higher order binary de Bruijn graphs in place, we see in the next chapter what the higher order graphs look like.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

Chapter III introduced the recursive process for constructing higher-order, odd-span, binary de Bruijn graphs. This chapter illustrates the results of the recursive generation and further details what was described previously. We also present certain additional properties of the graphs we construct in this chapter.

A. BUILDING THE GRAPHS

We first illustrate in Figures 4.1-4.4 how to construct the "building block", B_3. This illustration shows little regard to lengths B and C, as these are mostly involved in later iterations. Later in the chapter, particular attention will be paid to how changes in lengths B and C affect subsequent graphs.

STEP 1: As described in Chapter III, we first make four copies of the rotated graph B_1. For our illustration, length A is set to 2 inches and $\alpha = 15^o$. For purposes of labeling the copies and their respective nodes we append two high order bits, to each binary string of respective copies. The two appended bits are 00 for the first copy, 01 for the second copy, 10 for the third copy and 11 for the fourth copy. Note that different colors are also employed to distinguish between copies. For the purpose of our illustration, copy 0 is red, copy 1 is yellow, copy 2 is green and copy 3 is blue.

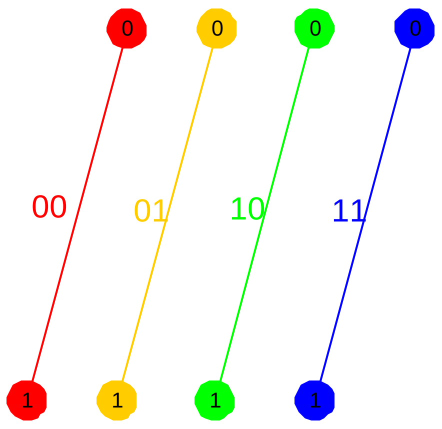


Figure 4.1. Four copies of B_1

STEP 2: Copy 0 is placed on a centerline CL_3 with its node 0, namely the node 000, on the centerline and the center point CP_3 is located according to parameter B. Copy 1 is placed with its node 0, namely node 010, on the centerline according to length C and its self-symmetry. That is, node 010 is on CL_3 as the node, 010, is self-symmetric. This results in Figure 4.2.

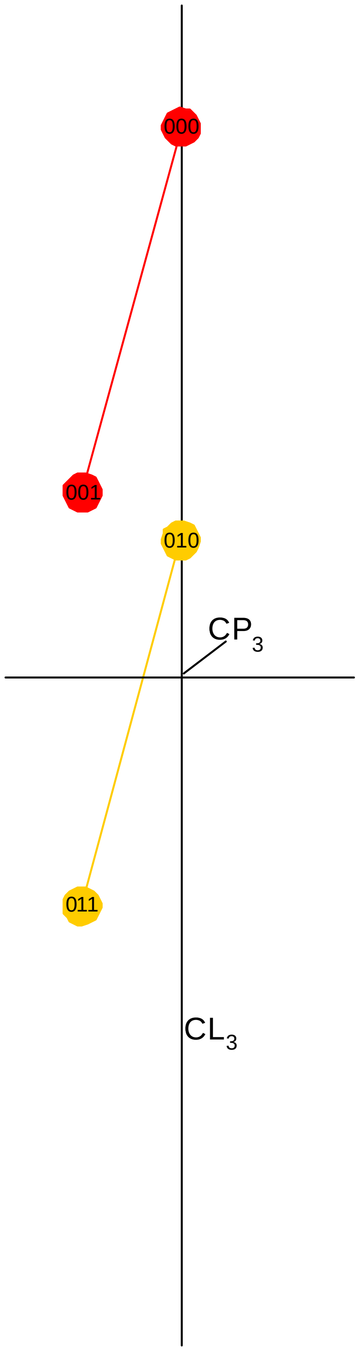


Figure 4.2. Placement of copies 0 and 1

STEP 3: We locate copies 2 and 3 according to top-bottom symmetry (property 2), left-right symmetry (property 1) and self-symmetry. Nodes 101 and 111 are self-symmetric, so appear on CL_3. Note that each of these nodes is node 1 in copy 2 and copy 3, respectively. The result appears in Figure 4.3.

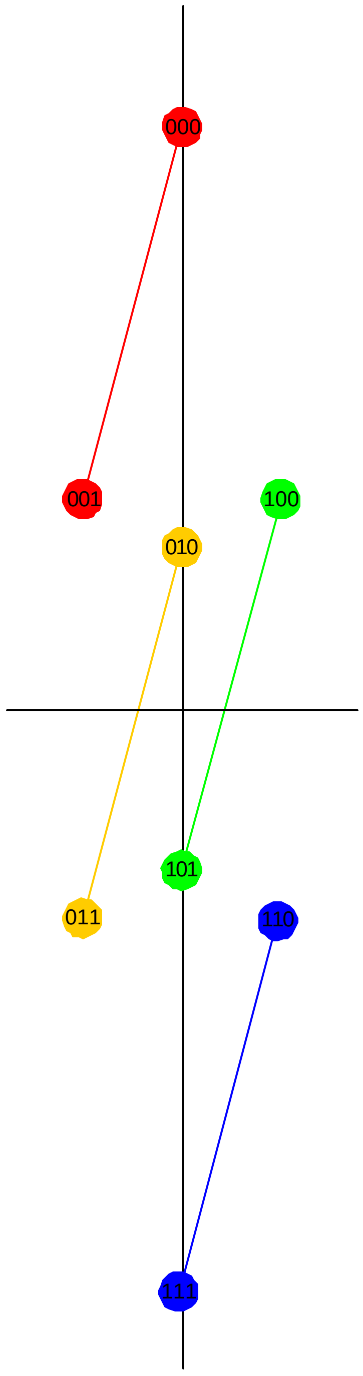


Figure 4.3. Placement of copies 2 and 3

STEP 4: We draw the outline shape, resulting in the skeleton of B_3 when we remove the construction lines. The final product is B_3, as appears in Figure 4.4, with most of its arcs suppressed and the CL_3 appearing.

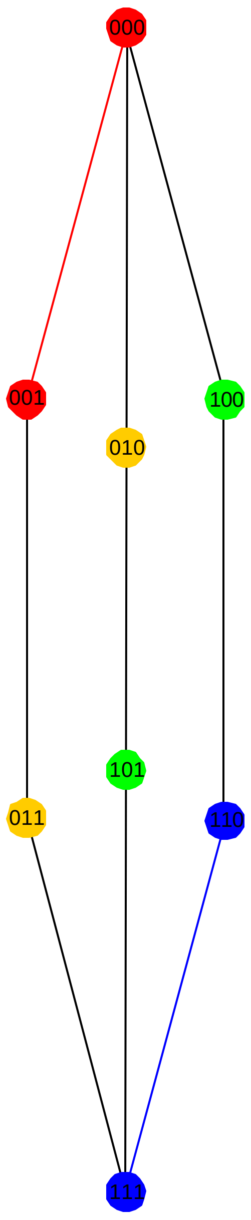


Figure 4.4. Recursively generated B_3

In creating the graph B_3 of Figure 4.4, the parameters A, B and C were each considered. However, once B_3 is constructed to be used as the basis for the additional recursive construction, these parameters are no longer used. The only parameter of importance for building subsequent graphs B_n now is the angle α. The symmetric properties (1) and (2) are the only guides we need for placement of the four copies we use after the building block B_3 is made. B_5 is drawn from four copies of B_3 in three steps, as illustrated in Figures 4.5-4.8. For this illustration B_3 is reduced in size so we could fit the graph on the page (A = 1.25 inches).

STEP 1: A copy of B_3 is rotated through the angle α. For this iteration $\alpha = 15^o$. Four copies of B_3 labeled 00, 01, 10, 11 are made. Notice in Figure 4.5 that in the center of each copy of B_3 are the two digits that we append when labeling the nodes in B_5. The labeling follows a similar pattern each time we employ it.

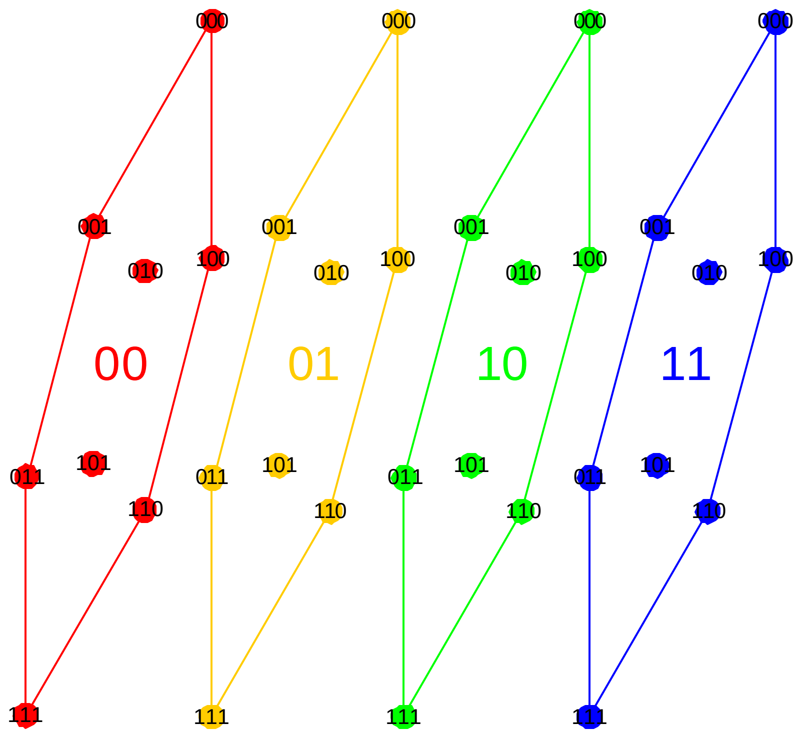


Figure 4.5. Four copies of B_3

STEP 2: We place the four labeled copies of B_3 on CL_5 according to symmetry properties (1) and (2). The centerline nodes for CL_5 are the self-symmetric nodes, namely 00000, 00100, 01010, 01110, 10001, 10101, 11011 and 11111. Figure 4.6 illustrates the placement of B_3 (red) and the positions of some of the other nodes of B_5 required by symmetry property (1) as they would appear in a symmetrically placed copy of B_3.

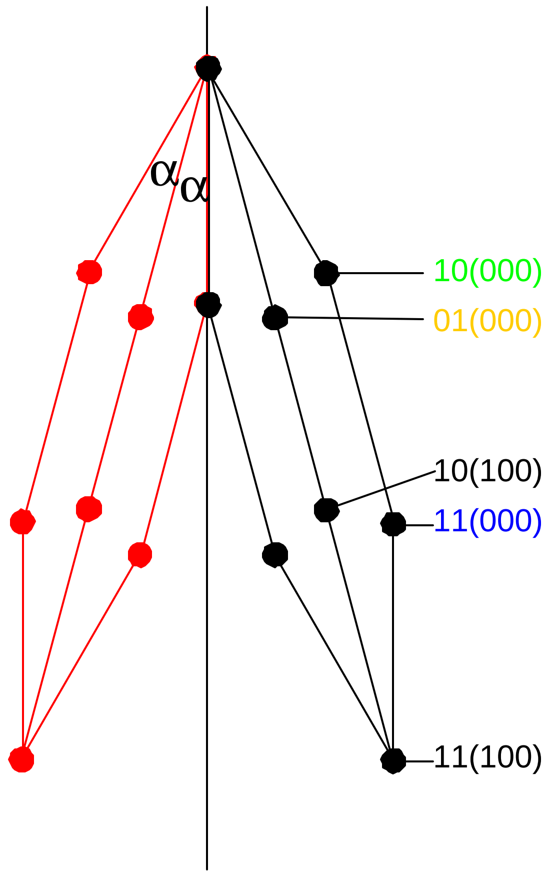
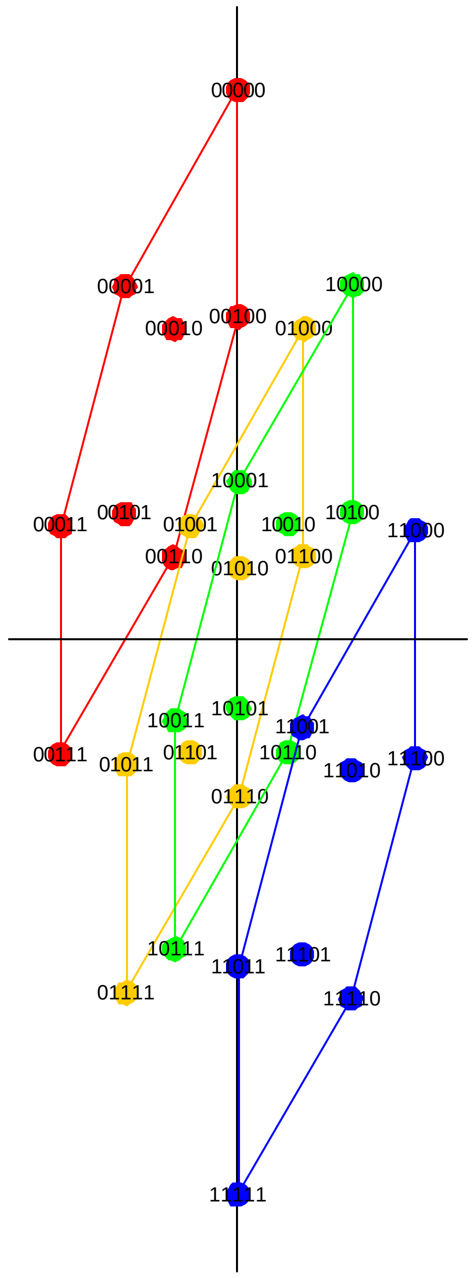


Figure 4.6. Illustration of Symmetry Property 1)

Notice the location of the nodes 10(000) and 01(000) and their colors. These are symmetrically placed opposite nodes 00(001) and 00(010), respectively. They are also used to place the second and third copies of B_3. Node 11(000), symmetrically placed opposite node 00(011), is used to place the fourth copy of B_3. Since α is the same angle for each copy, we are done when these four nodes are placed as the four copies of B_3 can then be located. We make the point that the other nodes on the centerline of the right (black) copy of B_3 are 00000, 01000, 10100 and 11100. Their eventual appearance as 00(00000), 00(01000), 00(10100) and 00(11100) on CL_7 is ensured by there appearance on CL_3 of this (black) copy! It follows that these nodes are on a line in B_5 from symmetry property (1) and their appearance as 000, 010, 101 and 111 on CL_3 in the respective locations. The drawing of the second, third and fourth copies, given their respective 000 node and the angle α, is easily seen in Figure 4.7.



Figures 4.7. Placement of copies 0, 1, 2 and 3

STEP 3: Finally, we draw the outline shape, the skeleton of B_5, and remove all other construction lines. The final graph is B_5. We retain the signature colors for clarity and suppress all the arcs for the same reason.

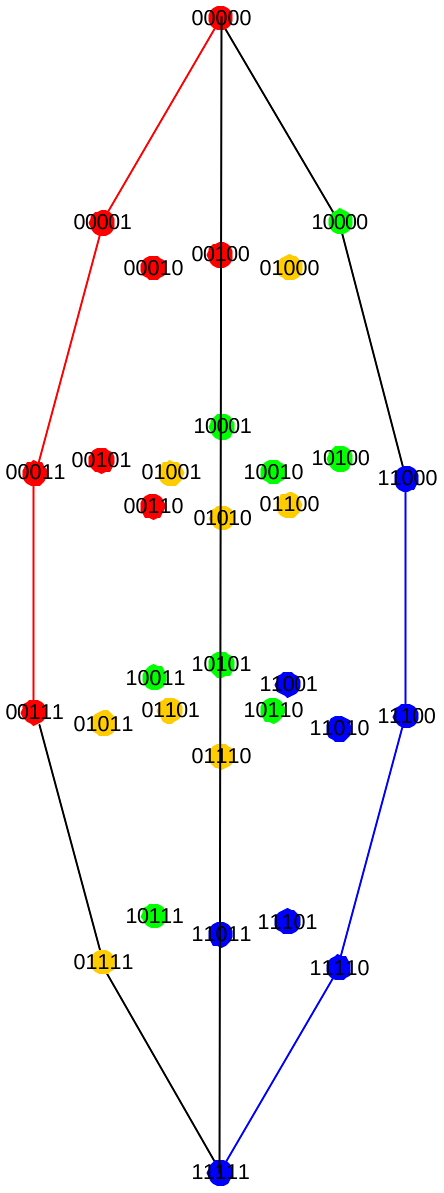


Figure 4.8. Recursively generated B_5

Just as we generated B_5, we also generate B_7, and so on. To ensure that the all the points are properly symmetrically placed, we consider the centerlines of each B_n, B_{n-2} and B_{n+2}.

We now address the issue of whether we like the building block we have created. If we use a traditionally shaped B_3, as shown in Figure 4.9,we create a B_5 and B_7 as also shown. Notice there results a reasonable-looking B_5, but the graph B_7 has some overlapping nodes that we don't like.

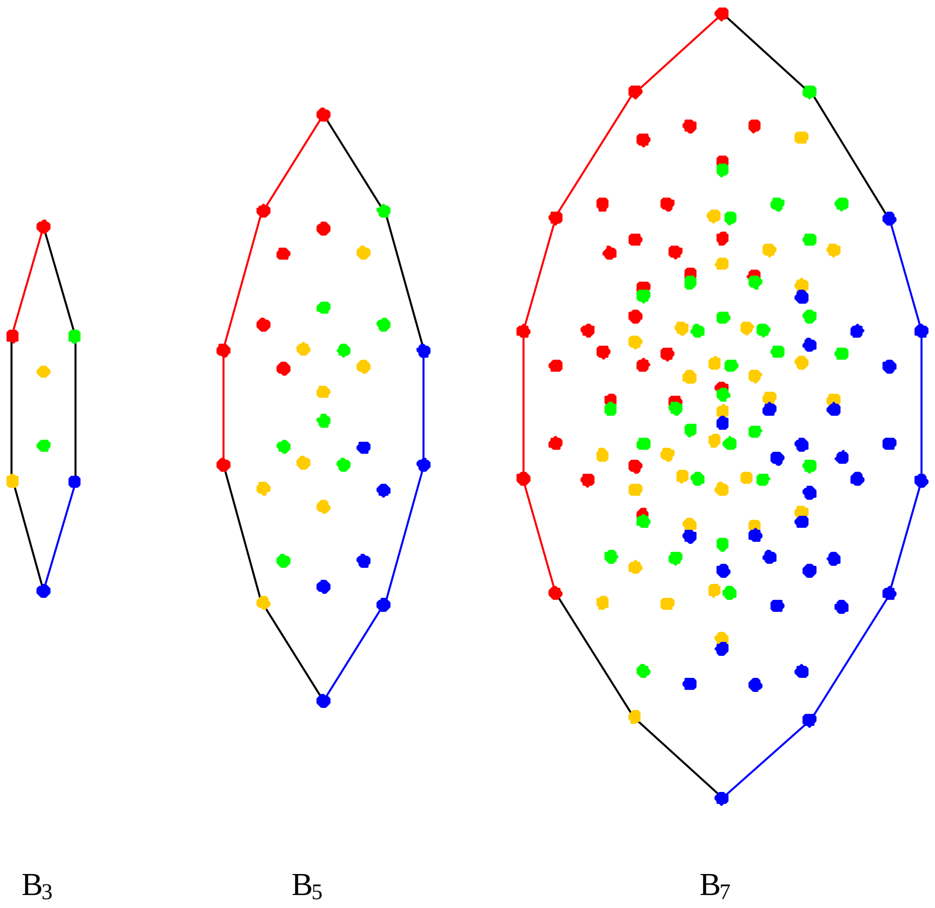


Figure 4.9. Traditional B_3, generating B_5 and B_7

Thus, the lengths B and C need adjustment. In Figure 4.10, length C has been adjusted so that node 010 is placed farther from the center point and lengths A and B remain the same. The result for this less traditional B_3 is tested on B_5 and B_7. The B_5 graph is reasonable but once again the B_7 graph is not.

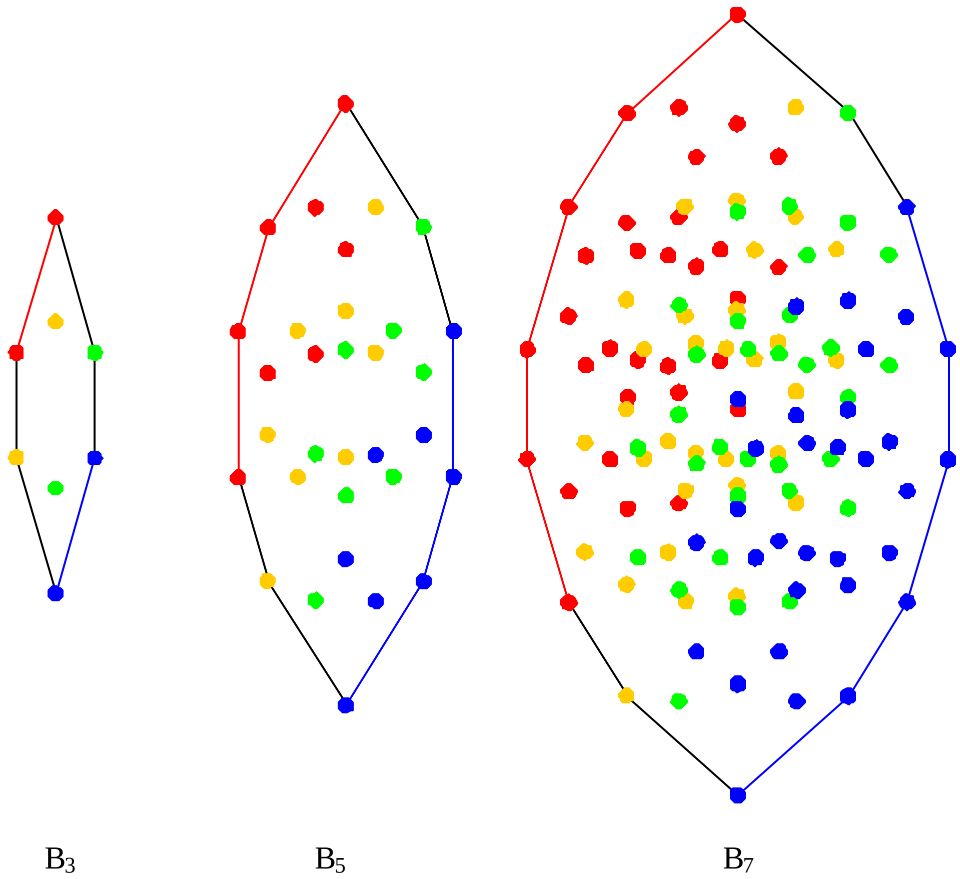


Figure 4.10. Non-traditional B_3, generating B_5 and B_7

By further adjustment, mainly to length C, we finally are able to produce a reasonable-looking B_7. The top center node, 010 of B_3 lies just below the line, between the nodes 001 and 100 as shown in Figure 4.11.

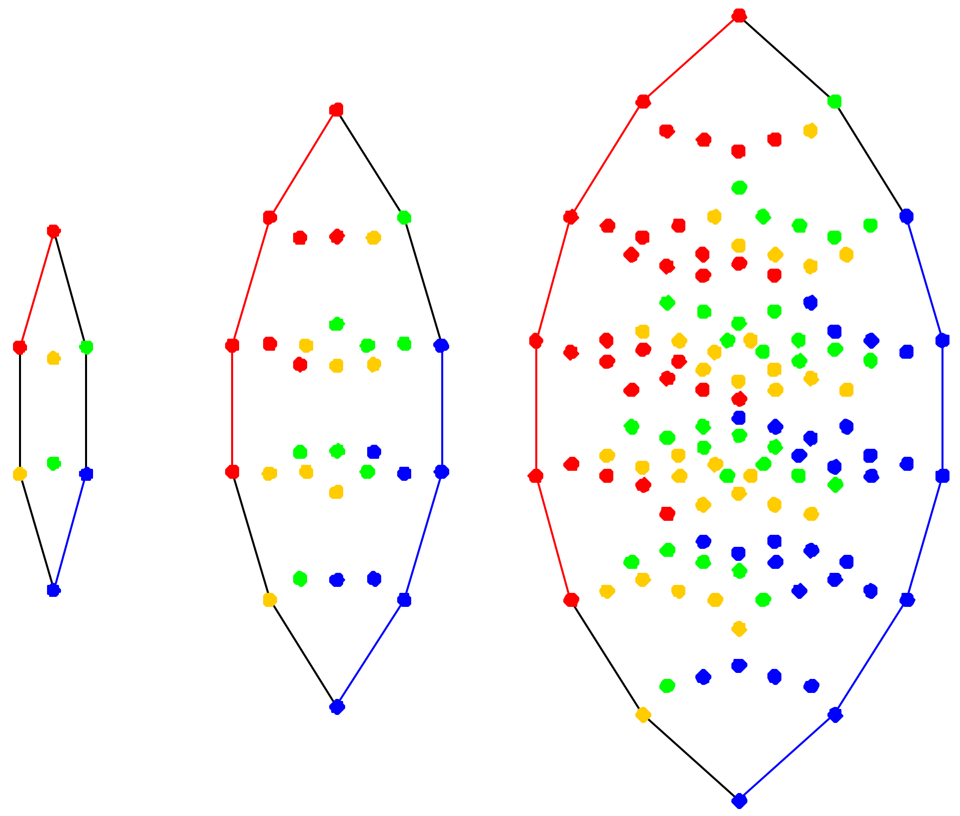


Figure 4.11. Model B_3, generating B_5 and B_7

So this model of B_3 yields a reasonably nice B_7 and is used further in the creation of the graph B_9 illustrated in the next section of this chapter. We also produce graph B_{11} from this model of B_3; this appears in Chapter V. The graphs B_{11} (Figure 5.3) in Chapter V reveal that further adjustment of not only length C but also the parameters α and B are necessary. As α gets smaller, pursuant to our desire to hold $\beta \leq 60^o$, C can get smaller by moving node 010 to be closer to the center point CP_3 and B can get longer and we still produce a reasonable looking B_7. Figure 4.12 shows the graph B_3 (and the respective B_5 and B_7) used in the construction of graphs B_{11} and B_{13}, (Figures 4.15 and 4.16) illustrated in the next section.

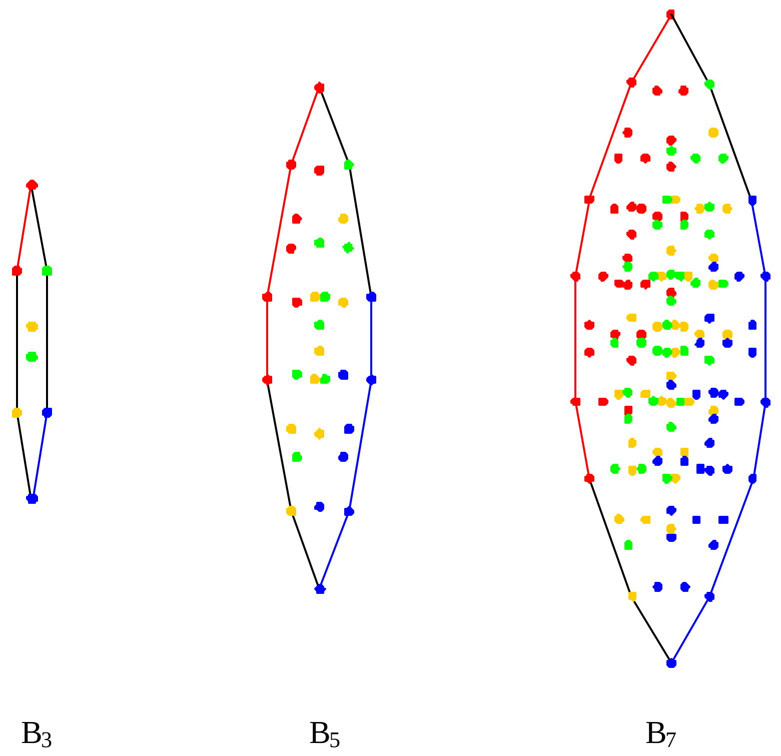


Figure 4.12. B_3 used to generate B_5-B_{13}

Thus, when determining parameters to adjust, we have to know the desired outcome graph B_n. In Chapter V we outline a scaling procedure that would allow us to ignore this step. Recall that we mentioned earlier that the nodes of the graphs would overlap eventually. As the span of the graph grows it is unavoidable that some overlapping will occur since the number of nodes grows exponentially and the space to fit them in is finite.

The $B_n \rightarrow B_{n+2}$ recursive technique is a smooth, simple construction for building odd higher-order binary de Bruijn graphs. First we construct the building block B_3 in four steps, then a three-step algorithm, using the recursive generation, is performed to produce B_{n+2} from B_n. This method proves to make the de Bruijn graph easily extendible. With this method, higher order graphs can be constructed and properties of these larger graphs can be discovered and developed.

B. PROPERTIES OF THE GRAPH

Through experimentation, some interesting properties were discovered about the graphs constructed. These properties address the importance of the labeling system we employ, the building block B_3 and how the centerlines CL_n shift as the span n grows.

The labeling system and color-coding we use make identifying specific nodes a simple exercise. In Figures 4.13 and 4.14, we demonstrate the ease of locating a particular node in B_9. Such a question would need to be considered if we were to draw the arcs $x \rightarrow 2x$ and $x \rightarrow 2x+1$ in the graph. Consider as an example the node numbered 294. Where is node 294 on the B_9 graph?

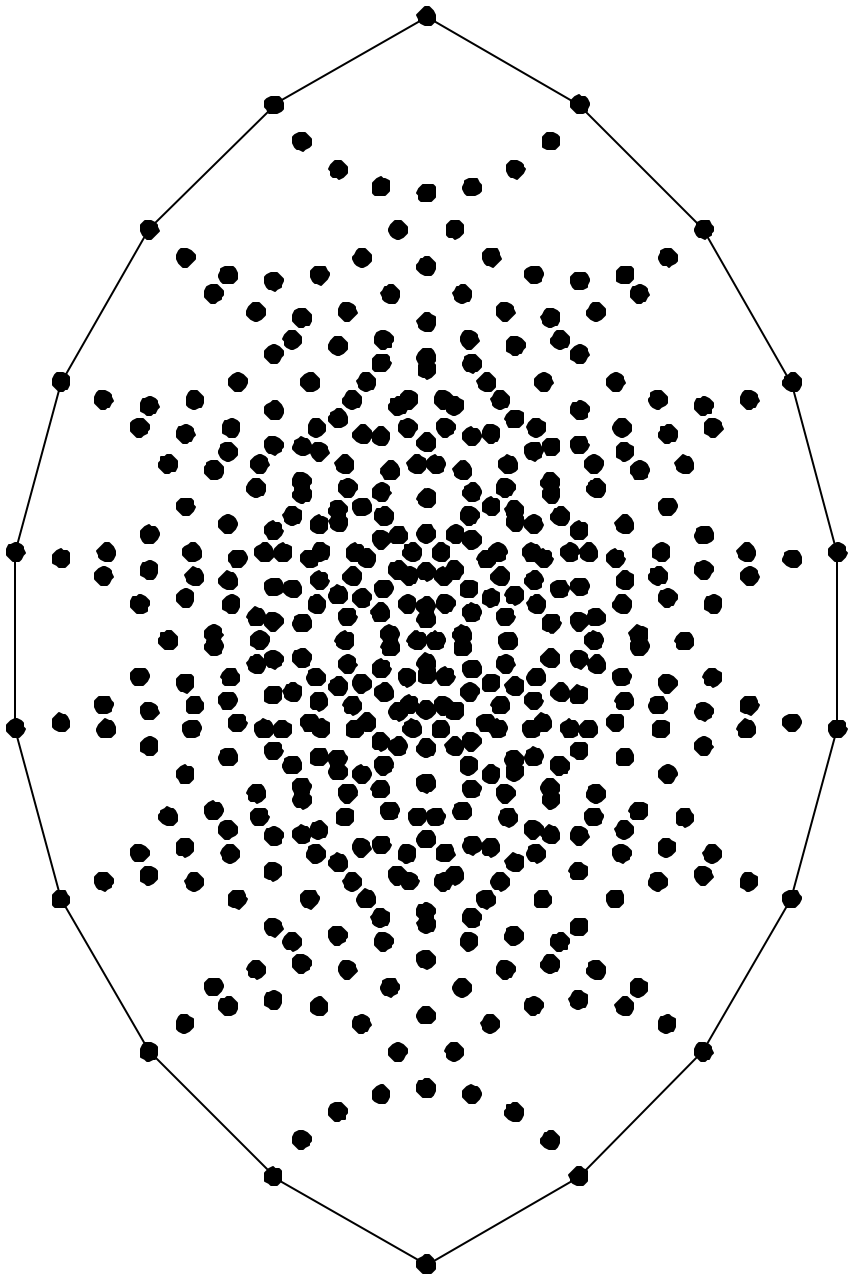


Figure 4.13. Recursively generated B_9

The binary representation of 294 gives a hint to its location. The binary representation for 294 is 100100110. We decompose the binary string. The first two bits, 10, represent copy 2 of the B_7 graph used to construct B_9. The next two bits, 01, represent copy 1 of the B_5 graph used to construct B_7. The next two bits, 00, represent copy 0 of the B_3 graph used to construct B_5. The final three bits 110 give the location on B_3. So 294 is in the third copy of the second copy of the first copy of the node 110 of B_3 all represented on the 9-graph. Note, some of the nodes would be colored initially as green, then yellow, then red. We only retain the last color they take on.

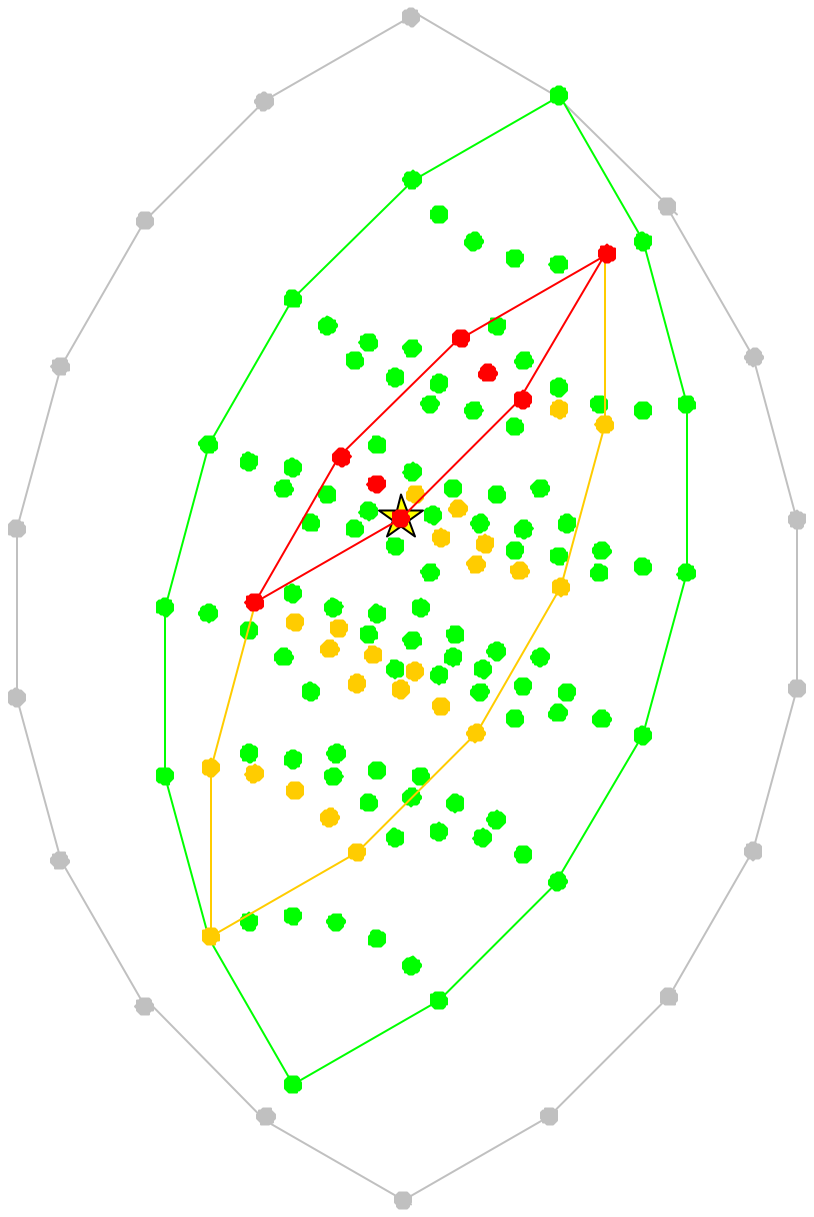


Figure 4.14. Location of node 294 on B_9

Another interesting property of the construction is its sparse appearance near the outer shell. There appears to be a relationship among these nodes. That relationship is related to the building block, B_3. The building block B_3 can be found in many guises throughout all the generated graphs. However one unique location of copies of B_3 is as they appear along the edges of B_n. As the graph grows, to span 11 and span 13, an interesting phenomenon occurs. The center of B_n becomes dense, but the region closer to the graph's outline maintains a sparse appearance as we noted. The nodes near the exterior can be viewed as a set of overlapping copies of B_3 around the edge. Consider the graphs in Figures 4.15 and 4.16, where copies of B_3 are highlighted in red. The analysis of this phenomenon can help us understand the density of nodes appearing in B_n. Note that the density of the nodes is becoming extreme in B_{11} and B_{13}. If we used smaller dots to indicate the nodes it would not appear so dense in the center. Nevertheless, the center of B_n is much more dense than the outer shell.

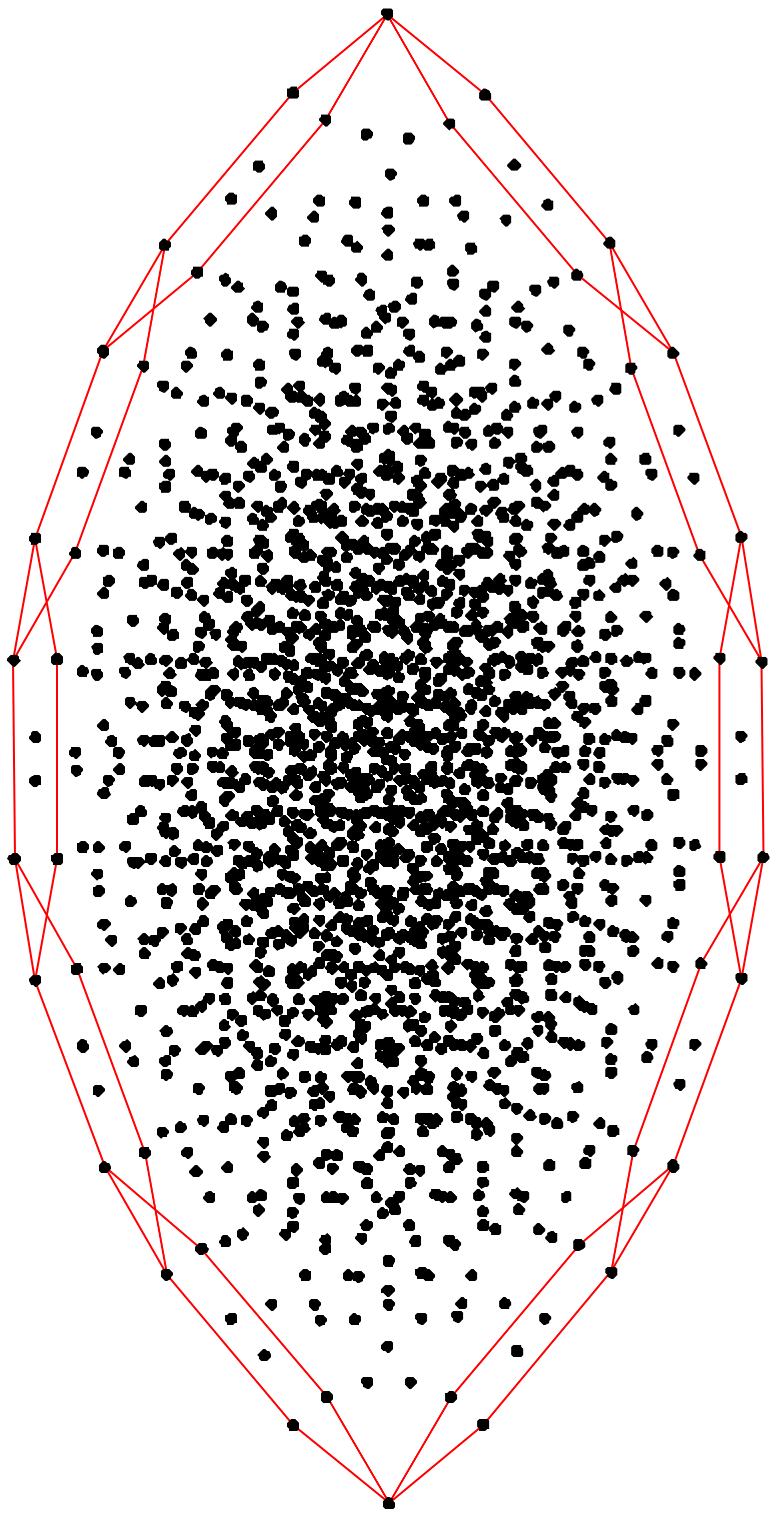


Figure 4.15. B_{11} with outlining copies of B_3

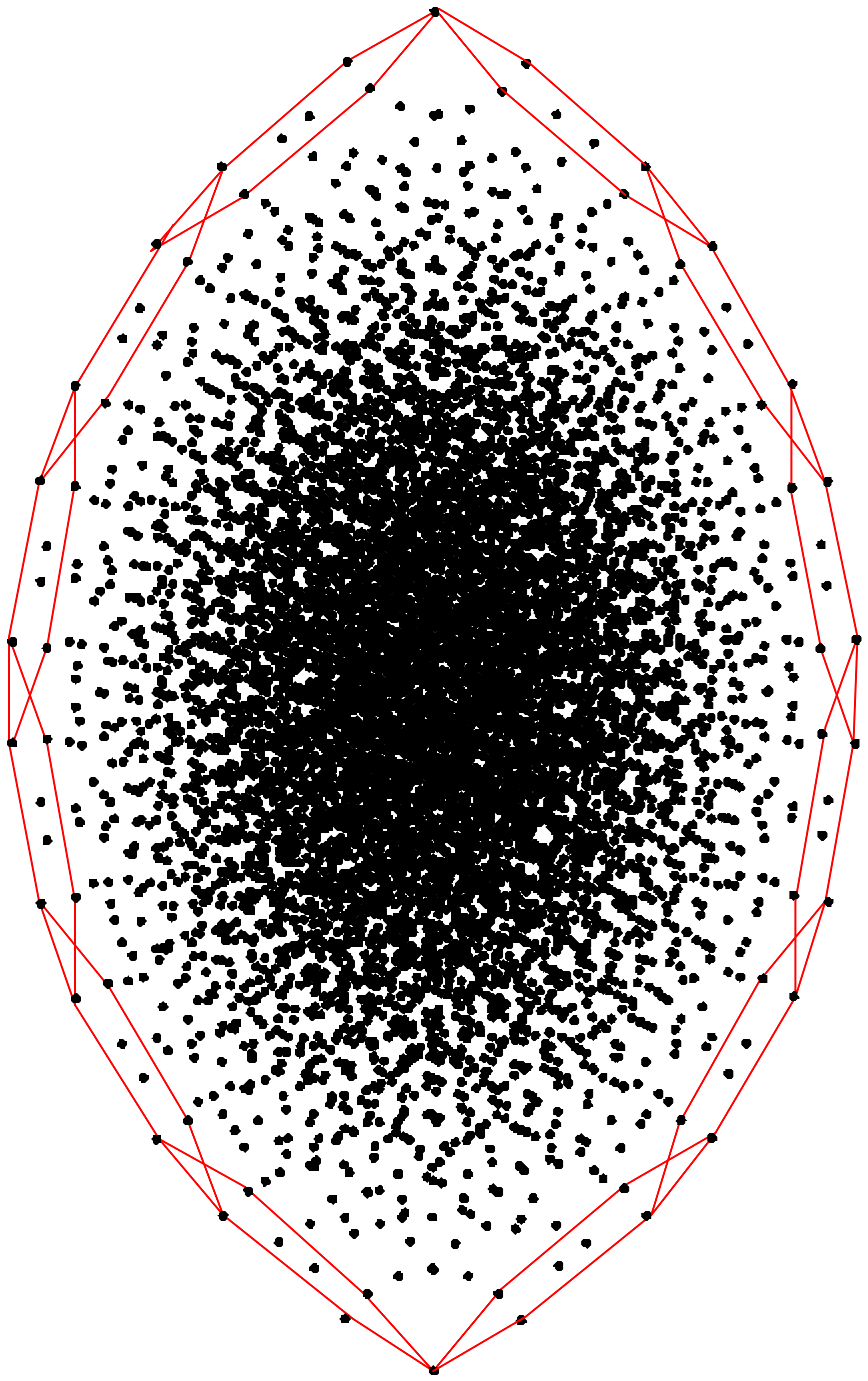


Figure 4.16. B_{13} with outlining copies of B_3

An additional property that we address involves properties of the centerlines of the graphs. From the recursive process for constructing B_{n+2} we use four copies of the previous graph B_n. The centerlines of the copies are used primarily to represent how the copies shift in each iteration. They also show that our construction rules are consistent, as we inferred earlier. These centerlines replicate the four copies that arise first in the design of B_3. If the four centerlines of B_n are reflected in B_{n+2} about CL_{n+2}, the nodes that they intersect on B_{n+2} are the nodes that will ultimately appear on CL_{n+4}. Namely, the dotted line in B_5 of Figure 4.17 is the reflected version (symmetry1) of copy 0 of CL_3 of B_3. The nodes on this gray line appear as the red nodes of the centerline CL_7 of B_7. We addressed this point when we discussed building B_5 in Section III.C. We also note that the centerlines of the second and third copies of B_n appear very close together in B_{n+2}. As n grows all four of these centerlines B_n become almost vertical and very close together. Figure 4.17 compares n = 3 to n = 5 and 7.

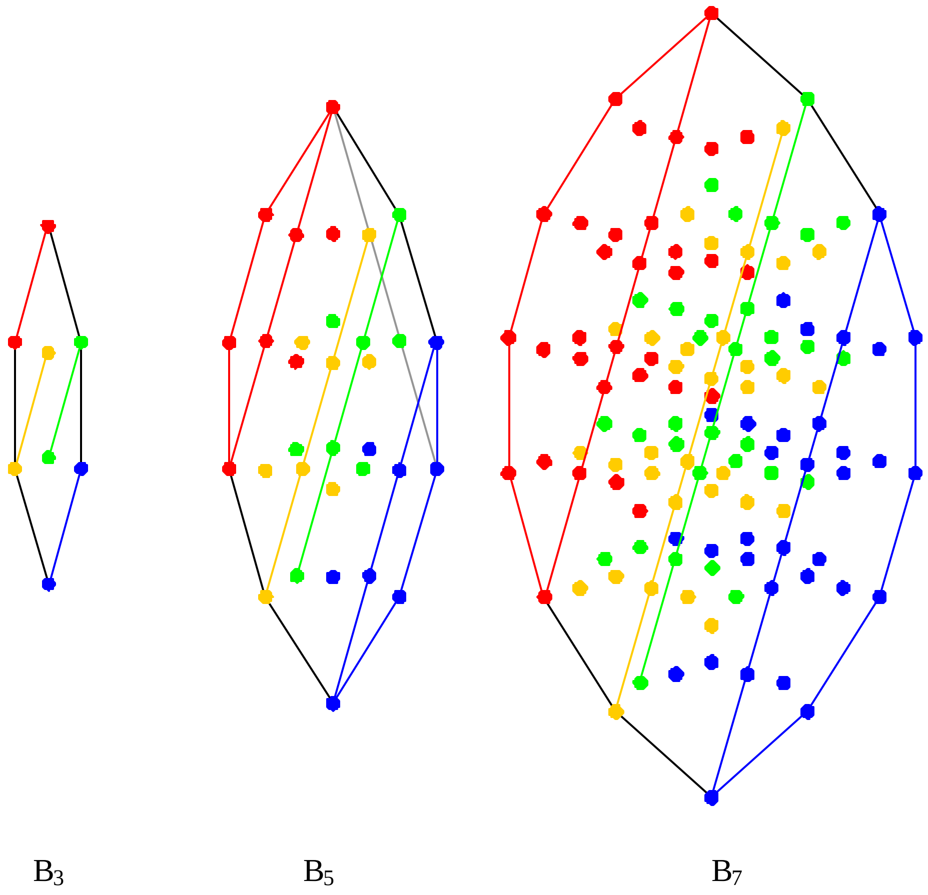


Figure 4.17. Centerlines on B_3, B_5 and B_7

Angle β increases with each iteration, and in turn the centerlines of the first and second copies of B_n draw closer together. As the recursion progresses and the graph continues to grow, the centerlines of these copies 1 and 2 will eventually overlap and ultimately change places. As this happens, β becomes distorted and the entire graph becomes indistinct. This forms the basis for our requirement that $\beta \leq 60^o$. Without requiring $\beta \leq 60^o$ this growth of the angle leads to the necessity to scale the graph. In Chapter V we address the importance of scaling and our thoughts on how it might be accomplished.

The properties presented here indicate that there are many more properties of the de Bruijn graph that remain to be discovered. We discuss some of the possibilities in the next chapter.

V. CONCLUSIONS AND RECOMMENDATIONS

In this chapter we address issues that arose when executing the recursive generation and some thoughts for the future research on this general topic.

A. SUGGESTIONS TO IMPROVE OUR IMPLEMENTATION

Two major issues of concern that arose during the experimental phase, efficiency of the process of building the graphs and our inability to scale the graphs.

The implementation of the recursive process of graph building was carried out manually. It was time-consuming to manually create each graph, and to copy, rotate and place the copies properly without the ability to set parameters automatically. Cleary, a computer program to build de Bruijn graphs by the recursive process is needed and clearly it is possible to produce such a program. With such a program, one could adjust the parameters in many different combinations and quickly view their effect. A program would enable us to consider even more models of building blocks to construct and test. We would then likely also discover more properties from the additional building blocks.

Not only does the process of building the graph need to be streamlined through a computer program but there also needs to be included in that program a method for scaling down graphs for larger orders. As we constructed the graphs, the required angle α on the building block B_3 was established in an ad hoc manner by choosing a size for a predetermined final graph. If the initial α was fixed and the span of a final graph was not predetermined, then a scaling factor would be necessary to construct graphs that obeyed the constraints given. For example if an initial value of the angle $\alpha = 20^o$; then the constructed B_7 has β ?at the maximum value of 60^o. The skeleton of the B_7 is shown in Figure 5.1.

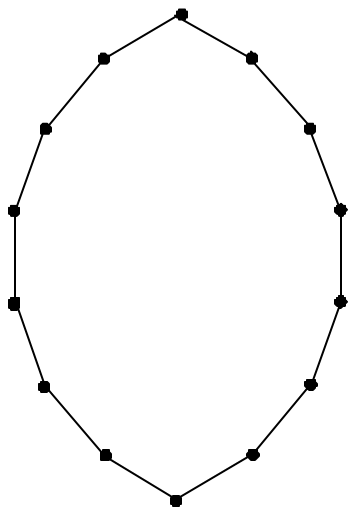


Figure 5.1. Outline of B_7 at $\alpha = 20^o$

Extending another iteration to B_9 shows that this instance of B_9 has $\beta = 80^o$. β obviously exceeds the constraint; see Figure 5.2. It is not yet problematic, but as we continue to B11 we see that B_{11} has a completely distorted β of 100^o; see Figure 5.3.

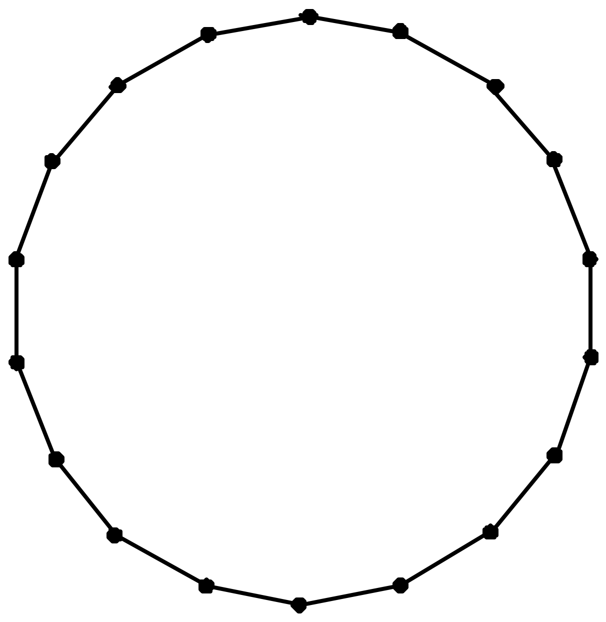


Figure 5.2. Outline of B_9 at $\alpha = 20^o$

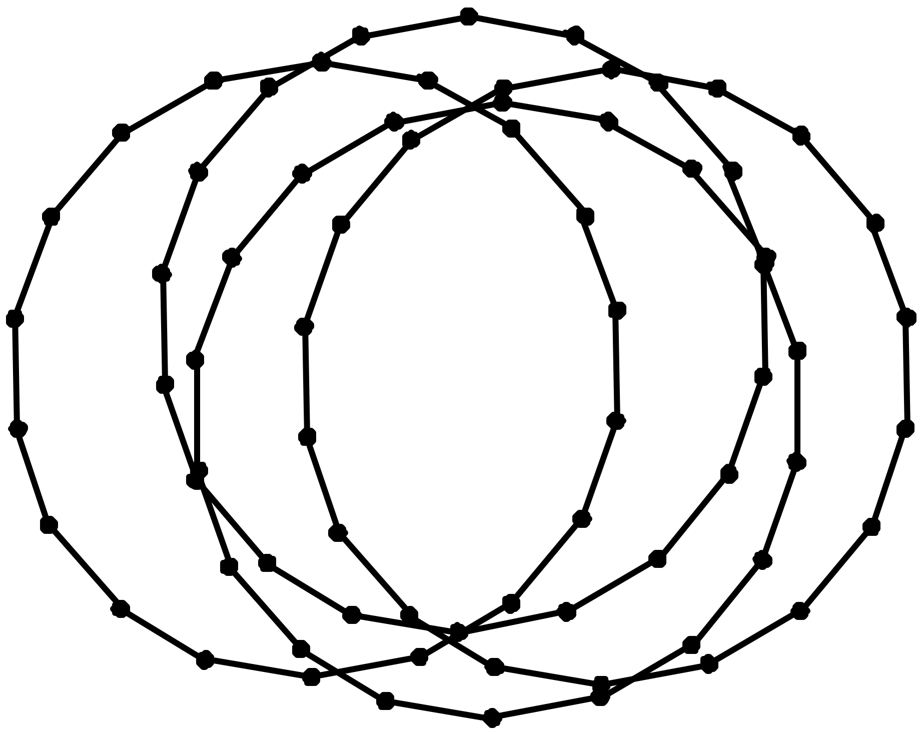


Figure 5.3. Outline of B_{11} with the 4 B_9 copies shown at $\alpha = 20^o$

This pumpkin shape is unacceptable. Continuing the process will allow β to exceed 360^o. The ability to scale the graphs during the recursive process is critical. We need a means of scaling the graph down after each iteration. It should be scaled by the same fraction each time so as to create an easy-to-follow algorithm within the recursive generation. In other words, a scaling algorithm should be embedded in the recursive construction of the graphs. The computer program to generate should include this feature.

B. FUTURE RESEARCH

Our goal when beginning this study was to produce de Bruijn graphs, through recursive generation, that were easy to construct and use. There were many techniques to choose from for the recursive process. We chose to consider the recursive process using

$B_n \rightarrow B_{n+2}$. We also presented only our work focusing on odd-to-odd order recursions, with B_1 forming a basis. We have some thoughts on the other recursive processes such as $B_n \rightarrow B_{n+1}$ and $B_n \rightarrow B_{n+3}$ and even to even recursion. There are different results to be achieved using these other techniques. For example, the even-span graphs have nodes lying on the line through CP_{2n} perpendicular to CL_{2n}. This is a little more cumbersome for our process. The transition from even to odd or odd to even changes the character of the graphs involved relative to this horizontal line. The issues are manageable but a little unwieldy. Additional nuances and properties are involved in completing other orders. For example, the basis for even to even construction may involve an additional parameter. These are further areas to be explored. The same recursive algorithm could be used but there are some differences in the outcomes.

We were able to build a “reasonable” looking B_7. We are intrigued to note that there appears to be some relationship between de Bruijn graphs and fractals. With the ability to scale the graphs, fractal behavior can be observed. In order to show that de Bruijn graphs are fractal, one would have to use a recursive process, show self-similarity and also show the graphs have fractional dimension. A recursive process creating the de Bruijn graphs has been demonstrated in the models. However, they do not appear self-similar in any traditional sense and currently have no fractional dimension that we have been able to demonstrate.

For future experimentation, a computer program will expedite the building process of de Bruijn graphs. The program should have the ability to reduce or expand (scale) the graph during the recursive process. This feature is important because the next iteration of this research is to discover how the other orders behave and verify whether de Bruijn graphs are fractal or not.

LIST OF REFERENCES

Berge, Claude, *The Theory of Graphs and Its Applications*, John Wiley, Publishing, 1962.

Bryant, Roy D. and Fredricksen, Harold, “Covering the de Bruijn Graph,” *Discrete Mathematics 89*, pp. 133-148, 1991.

de Bruijn, Nicolas G., “A combinatorial problem,” *Koninklijke Nederlands Akademie van Wetenschappen Proceedings*, Volume 49, pp.758-764, 1946.

Collins, O., Dolinar, S., McEliece, R., and Pollara, F., “A VLSI decomposition of the de Bruijn graph,” *Journal of ACM,* Volume 39, pp. 931-948, Oct. 1992.

Di Battista, Giuseppe, Eades, Peter, Tamassia, Roberto, and Tollis, Ioannis G., *Graph Drawing, Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.

Fredricksen, Harold, “A survey of full length nonlinear shift register cycle algorithms,” *SIAM Review 24*, Series #2, pp.195-221, 1982.

Fredricksen, Harold, “A new look at the de Bruijn graph,” *Discrete Mathematics 37/38*, pp. 193-203, 1992.

Golomb, Solomon W., *Shift Register Sequences*, Holden-Day Incorporated, 1967.

Good, I.J., “Normal recurring decimals,” *J. London Mathematics Society*, Volume 21, pp.169-172, 1946.

Johnson, D.M. and Mendelson, N.S., Planarity properties of the Good-de Bruijn graphs, *Combinatorial Structures and Their Applications*, pp. 177-183, Gordon and Breach, 1970.

Krahn, Gary W., *Double Eulerian Cycles on de Bruijn Graphs,* Ph.D. Dissertation, Naval Postgraduate School, Monterey, California, June 1994.

Lempel, Abraham, “On external factors of the de Bruijn graph,” *Journal of Combinatorics Theory*, 11, pp.17-27, 1971.

Liu, R.W. and Massey, J.L., *Proceedings of Third Annual Alberton Conference on Circuit and System Theory*, pp.73-81, Oct. 1965.

Mykkeltveit, Johannes, “A proof of Golomb’s conjecture for the de Bruijn graph,” *Journal of Combinatorics Theory*, Series A 13, pp. 40-45, 1971.

Rosen, Kenneth H., *Discrete Mathematics and Its Applications, Fourth Edition*, pp. 202-218, WCB/McGraw Hill, 1999.

Taylor, Herbert, “Drawing de Bruijn Graphs,” *The Mathemagician and Pied Puzzler*, pp.197-198, 2001.

INITIAL DISTRIBUTION LIST